

How Bash Processes Command Lines

Worked Example 1

In this Video...

- We will be visually working through the process together on some example commands

By the end, you will be able to:

- Mentally walk through how the shell will process a given command line
- Explain how each of the steps of shell operation are handled for a given command line

Let's Go!

Download Section Cheat Sheet



Initial Command Line

```
echo $name > $out
```

Parameter Definitions:

```
name="simon.smith"
```

```
out="output.txt"
```

Step 1: Tokenisation – Identify Unquoted Metacharacters

echo  \$name  > \$out

Step 1: Tokenisation – Find Words and Operators

echo \$name > \$out

Step 2: Command Identification

```
echo $name > $out
```

Step 3: Expansions – Brace Expansion (Stage 1)

```
echo $name > $out
```

There are no brace expansions

Step 3: Expansions – (Stage 2)

```
echo simon.smith > output.txt
```

Step 3: Expansions – Word Splitting (Stage 3)

```
echo simon.smith > output.txt
```

There is no word splitting, because the results of the expansions do not contain space, tab, or newline characters

Step 3: Expansions – Globbing (Stage 4)

```
echo simon.smith > output.txt
```

There is **no globbing**, because none of the words contain any **unquoted special pattern characters**

Step 4: Quote Removal

```
echo simon.smith > output.txt
```

There is no quote removal

Step 5: Redirections

```
echo simon.smith > output.txt
```

The diagram illustrates a shell command with standard output redirection. The command `echo simon.smith` is enclosed in a purple rectangular box. The word `echo` is underlined with a blue line, and `simon.smith` is also underlined with a blue line. A white arrow originates from the right side of the purple box, with the label `stdout` positioned above it. The arrow points to the text `output.txt`.

Up Next:

Worked Example 2